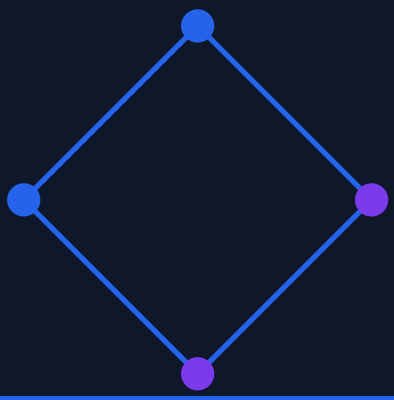
PMAI LAB

Mathematics • Python • Artificial Intelligence



LINEAR ALGEBRA FOR MACHINE LEARNING

Lecture 02

Vectors, Linear Combinations, Span, and Feature Spaces

Lecture focus Understanding vectors as data representations, combining vectors, describing the space they generate, and applying vector operations to features, embeddings, similarity, and prediction.

Dr. Hira Benish

Associate Professor of Mathematics

Student Learning Handout | Includes NumPy practice and review exercises

1 Lecture Overview

A vector is more than a list of numbers. It can represent an observation, a set of model parameters, a direction, or an embedding. Machine-learning systems convert complex objects such as students, customers, images, and words into vectors so that mathematical operations can be applied to them.

This lecture develops the concepts of vector operations, linear combinations, span, feature spaces, vector length, distance, and similarity. It also explains how these ideas appear in prediction and recommendation systems.

Learning Outcomes

After completing this lecture, students should be able to:

- represent real-world observations as vectors;
- perform vector addition, subtraction, and scalar multiplication;
- calculate dot products, vector norms, and Euclidean distances;
- construct and interpret linear combinations of vectors;
- explain the span of a set of vectors;
- interpret a dataset as points in a feature space;
- compute cosine similarity between vector representations; and
- implement vector operations using Python and NumPy.

1.1 Vectors Across Machine Learning

Application	Vector representation
Tabular prediction	Age, income, experience, and other features
Image recognition	Flattened or learned pixel features
Natural-language processing	Word, sentence, or document embeddings
Recommendation systems	User-preference and item-feature vectors
Neural networks	Inputs, weights, activations, and gradients
Clustering	Points compared using distance or similarity

Machine-Learning Connection

An **embedding** is a learned vector representation. Similar objects are designed to have nearby or similarly directed vectors, allowing a model to compare words, images, users, or products numerically.

2 Vectors and Their Structure

A vector in $\mathbb{R}^n$ is an ordered list of n real numbers:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n.$$

The number of entries is the vector's **dimension**. In machine learning, each entry generally corresponds to one feature.

2.1 Column and Row Vectors

A column vector has shape $n \times 1$, while a row vector has shape $1 \times n$:

$$\mathbf{x} = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}, \qquad \mathbf{x}^T = \begin{bmatrix} 2 & 4 & 6 \end{bmatrix}.$$

The transpose symbol T changes rows into columns and columns into rows. In NumPy, a one-dimensional array has shape `(n,)`, so its orientation depends on the operation in which it is used.

2.2 Equality and the Zero Vector

Two vectors are equal when corresponding entries are equal. The zero vector in $\mathbb{R}^n$ is

$$\mathbf{0} = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}^T.$$

2.3 Vector Operations

For $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and scalar c:

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} x_1 + y_1 \\ \vdots \\ x_n + y_n \end{bmatrix}, \qquad c\mathbf{x} = \begin{bmatrix} cx_1 \\ \vdots \\ cx_n \end{bmatrix}.$$

Example:

$$\mathbf{x} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}$$

gives

$$\mathbf{x} + \mathbf{y} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}, \qquad 2\mathbf{x} - \mathbf{y} = \begin{bmatrix} 5 \\ -1 \end{bmatrix}.$$

> Vectors can be added only when they have the same dimension. A shape mismatch in code often means that observations or features have been arranged inconsistently.

3 Length, Direction, and Distance

3.1 Euclidean Norm

The Euclidean norm measures the length of a vector:

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}.$$

For $\mathbf{x} = [3, 4]^T$,

$$\|\mathbf{x}\|_2 = \sqrt{3^2 + 4^2} = 5.$$

3.2 Unit Vector and Normalization

A unit vector has length 1. A non-zero vector can be normalized by

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}.$$

Normalization preserves direction while changing the vector's length to 1.

3.3 Euclidean Distance

The Euclidean distance between **x** and **y** is

$$d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2.$$

Distance is used in nearest-neighbour classification, clustering, anomaly detection, and similarity-based retrieval.

3.4 Feature Scaling Matters

Suppose a student vector contains age and annual family income. Income may have values in hundreds of thousands, while age may have values near 20. Without scaling, income will dominate Euclidean distance.

Practical Interpretation

Normalization changes the length of each vector. Standardization changes each feature using its mean and standard deviation. These operations have different purposes, even though both are often called "scaling."

Machine-Learning Connection

Distance-based methods such as k-nearest neighbours and k-means clustering are highly sensitive to feature scale. Vector geometry therefore influences model behaviour, not only mathematical notation.

4 Dot Product and Similarity

For vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, the dot product is

$$\mathbf{x}^T\mathbf{y} = x_1y_1 + x_2y_2 + \cdots + x_ny_n.$$

It produces a scalar and measures how strongly two vectors point in the same direction.

4.1 Geometric Form

$$\mathbf{x}^T\mathbf{y} = \|\mathbf{x}\|_2\|\mathbf{y}\|_2 \cos\theta,$$

where θ is the angle between the vectors.

- Positive dot product: broadly similar directions.
- Zero dot product: perpendicular directions.
- Negative dot product: broadly opposite directions.

4.2 Cosine Similarity

Cosine similarity compares directions while reducing the effect of magnitude:

$$\text{cosSim}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|_2\|\mathbf{y}\|_2}.$$

Its value lies between -1 and 1 for real non-zero vectors. Values closer to 1 indicate more similar directions.

Recommendation Example

Let a user's interests in technology, mathematics, and design be $\mathbf{u} = [5, 4, 1]^T$. Two course profiles are $\mathbf{c}_1 = [5, 5, 1]^T$ and $\mathbf{c}_2 = [1, 2, 5]^T$. Cosine similarity can be used to identify which course profile is better aligned with the user's interests.

Machine-Learning Connection

Modern search and retrieval systems compare query embeddings with document embeddings. Cosine similarity is one common way to rank vector representations by semantic closeness.

5 Linear Combinations

A **linear combination** of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is any vector of the form

$$c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_k\mathbf{v}_k,$$

where $c_1, \ldots, c_k$ are scalars.

For

$$\mathbf{v}_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \qquad \mathbf{v}_2 = \begin{bmatrix} 3 \\ 1 \end{bmatrix},$$

the combination $2\mathbf{v}_1 - \mathbf{v}_2$ is

$$2\begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} 3 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 3 \end{bmatrix}.$$

5.1 Weighted Prediction as a Linear Combination

Suppose normalized attendance, assignment performance, and quiz performance are

$$\mathbf{x} = \begin{bmatrix} 0.80 \\ 0.70 \\ 0.90 \end{bmatrix}, \qquad \mathbf{w} = \begin{bmatrix} 0.40 \\ 0.30 \\ 0.30 \end{bmatrix}.$$

The weighted score is

$$\mathbf{w}^T\mathbf{x} = 0.40(0.80) + 0.30(0.70) + 0.30(0.90) = 0.80.$$

This score is a linear combination of the student's features.

5.2 Matrix-Vector Form

For several observations stored as rows of X, the predictions can be calculated simultaneously:

$$\widehat{\mathbf{y}} = X\mathbf{w}.$$

Each entry of $\widehat{\mathbf{y}}$ is the dot product of one observation with the weight vector.

Machine-Learning Connection

Linear regression, logistic regression before activation, and neural-network layers all begin by forming weighted linear combinations of input features.

6 Span and Feature Spaces

The **span** of vectors $\mathbf{v}_1, \ldots, \mathbf{v}_k$ is the set of all their linear combinations:

$$\text{span}\{\mathbf{v}_1, \ldots, \mathbf{v}_k\} = \left\{ \sum_{i=1}^{k} c_i \mathbf{v}_i : c_i \in \mathbb{R} \right\}.$$

6.1 Geometric Examples

1. A single non-zero vector in $\mathbb{R}^2$ spans a line through the origin.
2. Two non-parallel vectors in $\mathbb{R}^2$ span the entire plane $\mathbb{R}^2$.
3. Two parallel vectors still span only one line because one repeats the direction of the other.
4. Three suitable, non-coplanar vectors in $\mathbb{R}^3$ span all of $\mathbb{R}^3$.

For example,

$$\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \qquad \mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

span $\mathbb{R}^2$ because every $[a, b]^T$ can be written as $a\mathbf{e}_1 + b\mathbf{e}_2$.

6.2 Feature Space

If a dataset has n features, every observation is represented as a point in an n-dimensional **feature space**. The dataset

$$X = \begin{bmatrix} 0.80 & 0.70 & 0.90 \\ 0.60 & 0.85 & 0.75 \\ 0.90 & 0.65 & 0.80 \end{bmatrix}$$

contains three observations in $\mathbb{R}^3$.

Column Space Interpretation

The columns of X generate its column space. A model of the form $X\mathbf{w}$ can produce only outputs lying in that column space. This observation becomes important when studying least-squares approximation.

Machine-Learning Connection

Feature engineering changes the vector representation and therefore changes the geometry of the feature space. A useful representation makes relevant patterns easier for a model to identify.

7 Python and NumPy Practice

7.1 Vector Operations

```python
import numpy as np

x = np.array([2.0, 1.0])
y = np.array([-1.0, 3.0])

print("x + y:", x + y)
print("2x - y:", 2 * x - y)
print("Dot product:", x @ y)
print("Length of x:", np.linalg.norm(x))
print("Distance:", np.linalg.norm(x - y))
```

7.2 Normalization and Cosine Similarity

```python
def normalize(v):
    length = np.linalg.norm(v)
    if np.isclose(length, 0):
        raise ValueError("The zero vector cannot be normalized.")
    return v / length

def cosine_similarity(a, b):
    return (a @ b) / (
        np.linalg.norm(a) * np.linalg.norm(b)
    )

u = np.array([5.0, 4.0, 1.0])
c1 = np.array([5.0, 5.0, 1.0])
c2 = np.array([1.0, 2.0, 5.0])

print("Normalized user:", normalize(u))
print("Similarity with course 1:", cosine_similarity(u, c1))
print("Similarity with course 2:", cosine_similarity(u, c2))
```

> Never normalize the zero vector because its norm is zero and division by zero is undefined. A robust function should check for this case explicitly.

8 Linear Combinations in Python

8.1 Constructing a Linear Combination

```
v1 = np.array([1.0, 2.0])
v2 = np.array([3.0, 1.0])

c1, c2 = 2.0, -1.0
result = c1 * v1 + c2 * v2

print("Linear combination:", result)
```

8.2 Testing Whether Vectors Span $\mathbb{R}^2$

Place the vectors as columns of a matrix. If the matrix has rank 2, its columns span $\mathbb{R}^2$.

```
V = np.column_stack((v1, v2))
rank = np.linalg.matrix_rank(V)

print("Matrix of vectors:\n", V)
print("Rank:", rank)
print("Spans R^2:", rank == 2)
```

8.3 Weighted Scores for Several Students

```
X = np.array([
    [0.80, 0.70, 0.90],
    [0.60, 0.85, 0.75],
    [0.90, 0.65, 0.80]
])

weights = np.array([0.40, 0.30, 0.30])
scores = X @ weights

print("Scores:", scores)
print("Highest-scoring student:", np.argmax(scores) + 1)
```

Machine-Learning Connection

The operation $X\mathbf{w}$ applies the same weighted rule to every observation. Vectorization replaces repeated manual calculations with one efficient matrix operation, a standard practice in data science and machine learning.

9 Review and Independent Practice

Practice Tasks

1. For $\mathbf{x} = [2, -1, 3]^T$ and $\mathbf{y} = [1, 4, 0]^T$, calculate $\mathbf{x} + \mathbf{y}$, $3\mathbf{x}$, $\mathbf{x}^T\mathbf{y}$, and $\|\mathbf{x}\|_2$.
2. Normalize $\mathbf{v} = [3, 4]^T$ and verify that the resulting vector has norm 1.
3. Determine whether $[1, 2]^T$ and $[2, 4]^T$ span $\mathbb{R}^2$. Support your answer using rank.
4. Express $[7, 1]^T$ as a linear combination of $[1, 0]^T$ and $[0, 1]^T$.
5. Calculate the cosine similarity between $[4, 3, 0]^T$ and $[5, 2, 1]^T$ using NumPy.
6. Create a dataset with four students and three normalized features. Select a weight vector and calculate all weighted scores using one matrix multiplication.
7. Explain why Euclidean distance can be misleading when features use very different measurement scales.

9.1 Key Takeaways

- Vectors represent observations, parameters, directions, and embeddings.
- Vector operations require compatible dimensions and careful shape handling.
- Norm measures length; distance measures separation; cosine similarity compares direction.
- A linear combination forms a new vector by weighting and adding existing vectors.
- Span describes every vector reachable through those linear combinations.
- Observations become points in a feature space, and the representation determines its geometry.
- Matrix-vector multiplication efficiently applies the same weighted rule to many observations.

9.2 Key Terms

Feature vector	Vector norm	Unit vector
Euclidean distance	Dot product	Cosine similarity
Linear combination	Span	Feature space
Embedding	Weight vector	Vectorization

Next lecture: Vector spaces, subspaces, linear independence, basis, and dimension.